Project Description: Car Rental System API using C#

The Car Rental System API provides functionality to manage a fleet of cars, including booking, availability checks, rental history, and user management. The system will be built using C#, Entity Framework (EF), and follows best practices for building RESTful APIs. Below are the tasks to be completed for each part of the system.

1. Models

- **Car Model**: Represents a car in the rental system.
    - **Properties**: Id, Make, Model, Year, PricePerDay, IsAvailable (boolean indicating availability)
- **User Model**: Represents a user who can rent cars.
    - **Properties**: Id, Name, Email, Password, Role (Admin or User)

2. Services

- **Car Rental Service**: Handles the core business logic for renting cars, including checking availability and calculating rental prices.
    - **Methods**: RentCar, CheckCarAvailability
- **User Service**: Handles user management, including user registration and authentication.
    - **Methods**: RegisterUser, AuthenticateUser (returns JWT token)

3. Repositories

- **Car Repository**: Manages data operations for the Car model.
    - **Methods**: AddCar, GetCarById, GetAvailableCars, UpdateCarAvailability
- **User Repository**: Manages data operations for the User model.
    - **Methods**: AddUser, GetUserByEmail, GetUserById

4. API Filters for Validation

- **Task**: Add basic validation for input data, such as ensuring required fields are present when creating or updating cars or users.
    - **Implementation**: Use built-in data annotations (e.g., [Required], [EmailAddress]) to validate models automatically.

5. Middlewares

- **Task**: Create a middleware to handle JWT token validation.
    - **Implementation**: The middleware should intercept requests and check if a valid JWT token is provided for protected endpoints. If not, return an unauthorized error.

6. Controllers for CRUD Operations

- **Car Controller**: Exposes API endpoints for managing cars.
    - **Endpoints**:
        - GET /cars: Get a list of available cars
        - POST /cars: Add a new car to the fleet
        - PUT /cars/: Update car details and availability
        - DELETE/cars/{id}: Delete the car details
- **User Controller**: Exposes API endpoints for user registration and authentication.
    - **Endpoints**:
        - POST /users/register: Register a new user
        - POST /users/login: Login and get JWT token

7. Notification Handling System

- **Task**: Implement a simple email notification system to notify users when their car booking is successful.
    - **Implementation**: Use a service like SendGrid to send an email when a car is successfully rented. Include basic details such as the car's make and model, rental duration, and the user's name. **[Explore this task]**

8. Authentication and Authorization using JWT

- **Task**: Implement JWT-based authentication for securing the API.
    - **Implementation**:
        - Users can register and log in to get a JWT token.
        - Secure endpoints like car rentals so only authenticated users can book cars.
        - Use role-based authorization to differentiate between normal users and admin users (e.g., only admins can add cars).

9. Testing using Postman

- **Task**: Test the API endpoints using Postman.
    - **Implementation**: Create Postman collections to test the following:
        - User registration and login (JWT token)
        - Viewing and renting cars
        - Admin adding and updating cars